IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application. No: 10/602,551	§	Examiner:	Dao, Thuy Chan
Filed: June 24, 2003	§	Group/Art Unit:	2192
Inventor(s):	§	Atty. Dkt. No:	5150-80201
Thomas A. Makowski, Rajesh Vaidya, Deborah E. Bryant, Brian M. Johnson	§		
Title: TASK BASED POLYMORPHIC GRAPHICAL PROGRAM FUNCTION NODES	§		

REQUEST FOR PRE-APPEAL BRIEF REVIEW

Dear Sir or Madam:

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request. This request is being filed with a Notice of Appeal. The review is requested for the reason(s) stated below.

Applicant is in receipt of the Advisory Action mailed May 29, 2008. Claims 69-92 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks. Please note that for brevity, regarding the claims, only the primary arguments directed to the independent claims are presented, and that additional arguments, e.g., directed to the subject matter of the dependent claims, will be presented if and when the case proceeds to Appeal.

Claims 69-92 were rejected under 35 U.S.C. 102(b) as being anticipated by Kudukoli (US Pat. Pub. No. 2001/0024211 A1). Applicant respectfully disagrees.

Applicant respectfully submits that Kudukoli fails to teach or suggest **associating the determined graphical program code with the node, wherein, when the node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**, as recited in claim 69.

Cited Figure 4 and paragraphs [0100] – [0113] are directed to a method for programmatically creating a graphical program, specifically, via creation of a graphical program generation (GPG) program, typically by a user, after which the GPG program is executed. At runtime, the GPG program receives program information (via node inputs) specifying functionality for a new graphical program, and in response to this program information, the GPG program programmatically generates the new graphical program (or graphical program portion) implementing the specified functionality.

Cited Figure 23 and [0273] – [0275] disclose a graphical program that includes a user interface panel and a block diagram coupled to the user interface panel, where the graphical program was generated by the GPG program. Cited [0120] discusses the fact that the GPG program may be associated with a program or application, such as a graphical program development environment, that may directly aid the user in creating a new graphical program. Cited [0020] discloses that the GPG program may be constructed using any of various programming technologies or techniques, and may be a text-based program and/or a graphical program. Cited Figure 6 and [0136] – [0140] discuss the GPG program generating a graphical program in response to initial program information, then modifying the graphical program based on subsequent program information.

Applicant respectfully notes that:

1) the GPG program generates graphical program code for a *new* graphical program, not for itself. For example, the cited New VI Object Reference node in the GPG program is configured to create a new VI object (e.g., based on a VI object style input value), then when the GPG program is run, this node executes and creates a new VI object of the specified style in a *new graphical program*; and

2) the program code that creates the specified new VI object at runtime is inherently already part of the New VI Object Reference node, and, per 1), any graphical program code generated by the New VI Object Reference node belongs to the new graphical program.

Thus, in Kudukoli, there is no *associating* determined code with the New VI Object Reference node. Nowhere does Kudukoli mention or even hint at determining graphical program code for a node based on user-selection of displayed functions for the node, and *associating the determined graphical program with the node, where when the node executes, the determined graphical program code executes to perform the function.*

In fact, Kudukoli nowhere describes associating determined graphical program code with a node that is already displayed in a graphical program at all. The only associating Kudukoli discusses is between the GPG program and the received program information ([0037], [0147]), between the GPG program and a program or application that aids the user in creating the GPG program or a new graphical program, e.g., a development environment or application ([0022] and elsewhere), and between a new graphical program and a new programming environment ([0023] and elsewhere).

Thus, Applicant submits that even if Kudukoli's New VI Object Reference node code that generates a new VI object for new graphical program were considered to be Applicant's claimed "determined graphical program code", this code (of the New VI Object Reference node) inherently belongs to the New VI Object Reference node, and thus no *associating* of this code with the node is performed. Thus, the Advisory Action's assertion that Kudokoli teaches associating code for generating the waveform chart (or random number generator or stop button) with the node is incorrect, since this

code already belongs to the node. The Examiner has improperly read this claimed feature into Kudokoli without presenting any evidence to support the feature. Cited [0278]-[0279] and [0282] (or Kudukoli in general) in no way disclose this feature, nor does Kudokoli indicate anywhere that this code is graphical program code. Applicant respectfully requests that the Examiner particularly describe and explain where these features may be found in Kudokoli.

Similarly, any new VI object created by the New VI Object Reference node belongs to the new graphical program, and its code is neither associated with the New VI Object Reference node, nor executed when the New VI Object Reference node executes; rather, the new VI object is created by and when the New VI Object Reference node is executed. Applicant submits that in the Advisory Action, the Examiner appears to be blurring the distinction between the inherent code of the New VI Object Reference node that creates new nodes/objects for the new graphical program, and the created nodes/objects themselves, which is improper and incorrect.

Thus, for at least the reasons provided above, Applicant submits that Kudukoli fails to teach or suggest these features and limitations of claim 69, and so claim 69 and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claim 77 includes similar limitations as claim 69, and so the above arguments apply with equal force to this claim. Thus, for at least the reasons provided above, Applicant submits that claim 77 and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Regarding independent claim 85, nowhere does Kudukoli disclose **determining a second node based on the selected function, wherein the second node comprises a graphical representation of an implementation of the selected function, and wherein the second node comprises graphical program code executable to provide functionality in accordance with the selected function**.

Cited [0022] describes associating the GPG program with a program or application that aids the user in creating the GPG program or a new graphical program, e.g., a development environment, application, or wizard, but makes no mention of determining a second node based on a function selected by a user for a first node in a graphical program, where the second node is executable to perform the selected function. Note, for example, that Kudukoli's New VI Object Reference Node generates a new VI object based on inputs to the node, i.e., the input, e.g., user selection of object style, specifies what object the New VI Object Reference Node will create when executed. The new VI object is *not* executable to perform the selected functionality, that is, the creation functionality that generates the new VI object. Cited [0029] discusses dependence of the functionality of Kudukoli's generated graphical program on both received information and the GPG program. Again, this citation makes no mention of determining a second node based on a function selected by a user for a first node in a graphical program,

where the second node is executable to perform the selected function. Finally, cited [0030] describes the "typical case" where "the implementation of the graphical program code is determined mainly or entirely by the GPG program, although the received information may influence the manner in which the GPG program generates the code, or the GPG program may receive separate information influencing the code generation." More specifically, this citation discloses the GPG program translating an existing graphical program to a new graphical program.

Applicant submits that in the above citations, and Kudukoli in general, no description is provided of determining a second node based on user-selection of functionality for a first node in a graphical program, where the second node is executable to perform the selected functionality. The Advisory Action's assertion that "the waveform control/node (the second node) has replaced the New VI Object Reference Node (the first node) by having the same icon/appearance with the New VI Object Reference Node but with the determined graphical program code associated with 'waveform chart' [*sic*]" is incorrect. Kudukoli nowhere mentions a "waveform control/node", and, as noted above, the code that creates the waveform chart is *inherent* in the New VI Object Reference node and so no *associating* is performed, and furthermore, is nowhere described as graphical program code. Additionally, the waveform control is not a graphical program node, but rather a GUI element, and the created waveform chart user interface node is in the new graphical program, and thus does not replace the New VI Object Reference node. Similarly, the cited "random number generator control/node" ([0282]), which is actually referred to as "the random number generator function" node, the cited wait function node, and the cited NOT Boolean node, are all created (by the New VI Object Reference node) and inserted into the *new* graphical program, and thus do not replace the New VI Object Reference node. Again, Applicant submits that in the Advisory Action, the Examiner appears to be blurring the distinction between the inherent code of the New VI Object Reference node that creates new nodes/objects for the new graphical program, and the created nodes/objects themselves, which is improper and incorrect–the created nodes do not replace the New VI Object Reference node.

Thus, the cited art does not teach or suggest these claimed features.

Additionally, nowhere does Kudukoli disclose **replacing the node in the graphical program with the second node, wherein, when the second node in the graphical program executes, the determined graphical program code executes to provide the functionality in accordance with the selected function**, as recited in claim 85

Cited Figure 22 illustrates how a user may choose a value for the style input by selecting from a hierarchical menu, e.g., invoked with respect to a New VI Object Reference node. Note that once configured, the New VI Object Reference node executes in a GPG program to create a new VI object for or in *another graphical program*, as discussed above. The created object does not replace the New VI

Object Reference node in the GPG program. In other words, no node is replaced by the created VI object generated by the New VI Object Reference node.

Paragraph [0225] discusses the VI object class input to an Upcast Reference node, which specifies a class to which the Upcast Reference node will cast a VI object reference. Paragraph [0230] discusses the VI object class input to a Downcast Reference node, which specifies a class to which the Downcast Reference node will cast a VI object reference. Neither of these citations mentions or even hints at replacing a node in a graphical program with a second node that executes determined graphical program code as claimed.

Applicant has reviewed Kudukoli closely, and respectfully submits that no mention is made in the entire reference of performing Applicant's claimed node replacement in a graphical program based on user-selected functionality for the node.

Thus, or at least the reasons provided above, Applicant submits that Kudukoli fails to teach or suggest these features and limitations of claim 85, and so claim 85 and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claim 89 includes similar limitations as claim 85, and so the above arguments apply with equal force to this claim. Thus, for at least the reasons provided above, Applicant submits that claim 89 and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested. If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-80201/JCH.

Also filed concurrently is the following item: Notice of Appeal.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Date: 2008-06-10 JCH/MSW